Exploratory Test Styles

These slides are distributed under the Creative Commons License.

In brief summary, you may make and distribute copies of these slides so long as you give the original author credit and, if you alter, transform or build upon this work, you distribute the resulting work only under a license identical to this one.

For the rest of the details of the license, see http://creativecommons.org/licenses/by-sa/2.0/legalcode.

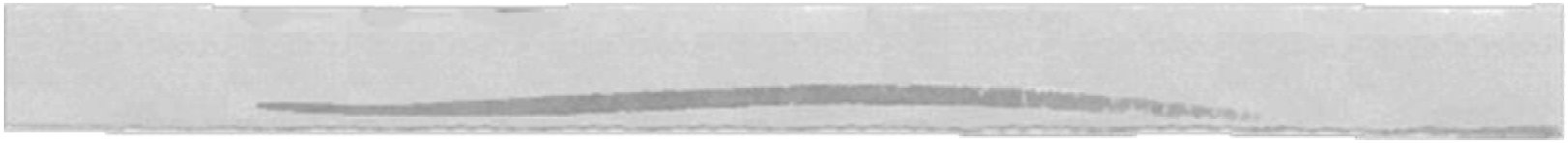
Styles of Exploration Outline

1. **Introduction**
2. **Test Management and Techniques**
3. **ET Planning, Exec. and Documentation**
4. **ET Styles**
5. **ET Management**

- **Hunches**
- **Models**
- **Examples**
- **Invariances**
- **Interference**
- **Error handling**
- **Troubleshooting**
- **Group Insight**
- **Specifications**

©2002 Amland Consulting

4. Exploratory Test Styles



Skills and Techniques

©2002 Amland Consulting

"In the fields of observation, chance favors only those minds which are prepared."

Louis Pasteur

©2002 Amland Consulting

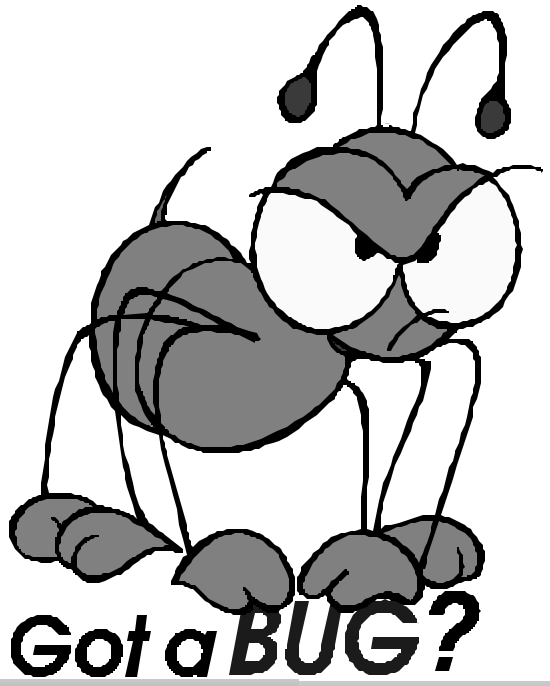
Approaches / Styles of ET

- At the heart of all ET styles:
  - *Questions* and *Questioning Skills*
- Characterize the styles with respect to each other:
  - Do they focus on:
    - Method of questioning?
    - Method of describing or analysing the product?
    - The details of the product?
    - The patterns of use of the product?
    - The environment in which the product is run?
  - To what extent would this style benefit from group interaction?

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration

- Hunches
- Models
- Examples
- Invariances
- Interference
- Error Handling
- Troubleshooting
- Group Insight
- Specifications



From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration:

Hunches

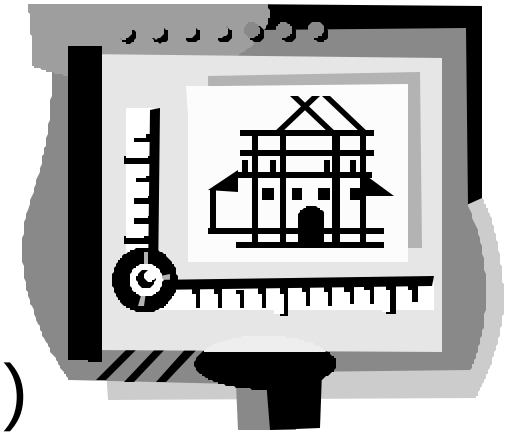
- ”Random”
- Questioning
- Similarity to previous errors
- Following up gossip and predictions
- Follow up recent changes

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration:

Models

- Architecture diagrams
- Bubble diagrams
- Data relationships
- Procedural relationship
- Model-based testing (state matrix)
- Requirements definition
- Functional relationship (for regression testing)
- Failure models



From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration: Models

Architecture Diagrams

- Work from a high level design (map) of the system
  - Pay primary attention to interfaces between components or groups of components. We're looking for cracks that things might have slipped through
  - What can we do to screw things up as we trace the flow of data or the progress of a task through the system?
- You can build the map in an architectural walkthrough
  - Invite several programmers and testers to a meeting. Present the programmers with use cases and have them draw a diagram showing the main components and the communication among them. For a while, the diagram will change significantly with each example. After a few hours, it will stabilize.
  - Take a picture of the diagram, blow it up, laminate it, and you can use dry erase markers to sketch your current focus.
  - Planning of testing from this diagram is often done jointly by several testers which understand different part of the system.

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration: Models

Bubble (Reverse State) Diagram

- To trouble shoot a bug, a programmer will often work the code backwards, starting with the failure state and reading for the state that could have led to it (and the states that could have led to those).
- The tester imagines a failure instead, and asks how to produce it.
  - Imagine the program being in a failure state. Draw a bubble.
  - What would have to have happened to get the program here? Draw a bubble for each immediate precursor and connect the bubbles to the target state.
  - For each precursor bubble, what would have happened to get the program there? Draw more bubbles.
  - More bubbles, etc.
  - Now trace through the paths and see what you can do to force the program down one of them.

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Data Relationship

- Pick a data item
- Trace its flow through the system
- What other data items does it interact with?
- What functions use it?
- Look for inconvenient values for other data items or for the functions, look for ways to interfere with the function using this data item

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration: Models

Procedural Relationships

- Pick a task
- Step by step, describe how it is done and how it is handled in the system (to as much detail as you know)
- Now look for ways to interfere with it, look for data values that will push it toward other paths, look for other tasks that will compete with this one, etc.

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration: Models

Functional Relationships

A model (what you can do to establish a strategy) for deciding how to decide what to regression test after a change:

1. Map program structure to functions.
   - This is (or would be most efficiently done as) a glass box task. Learn the internal structure of the program well enough to understand where each function (or source of functionality) fits
2. Map functions to behavioral areas (expected behaviors)
   - The program misbehaved and a function of functions were changed. What other behaviors (visible actions or options of the program) are influenced by the functions that were changed?
3. Map impact of behaviors on the data
   - When a given program behavior is changed, how does the change influence visible data, calculations, contents or data files, program options, or anything else that is seen, heard, sent, or stored?

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

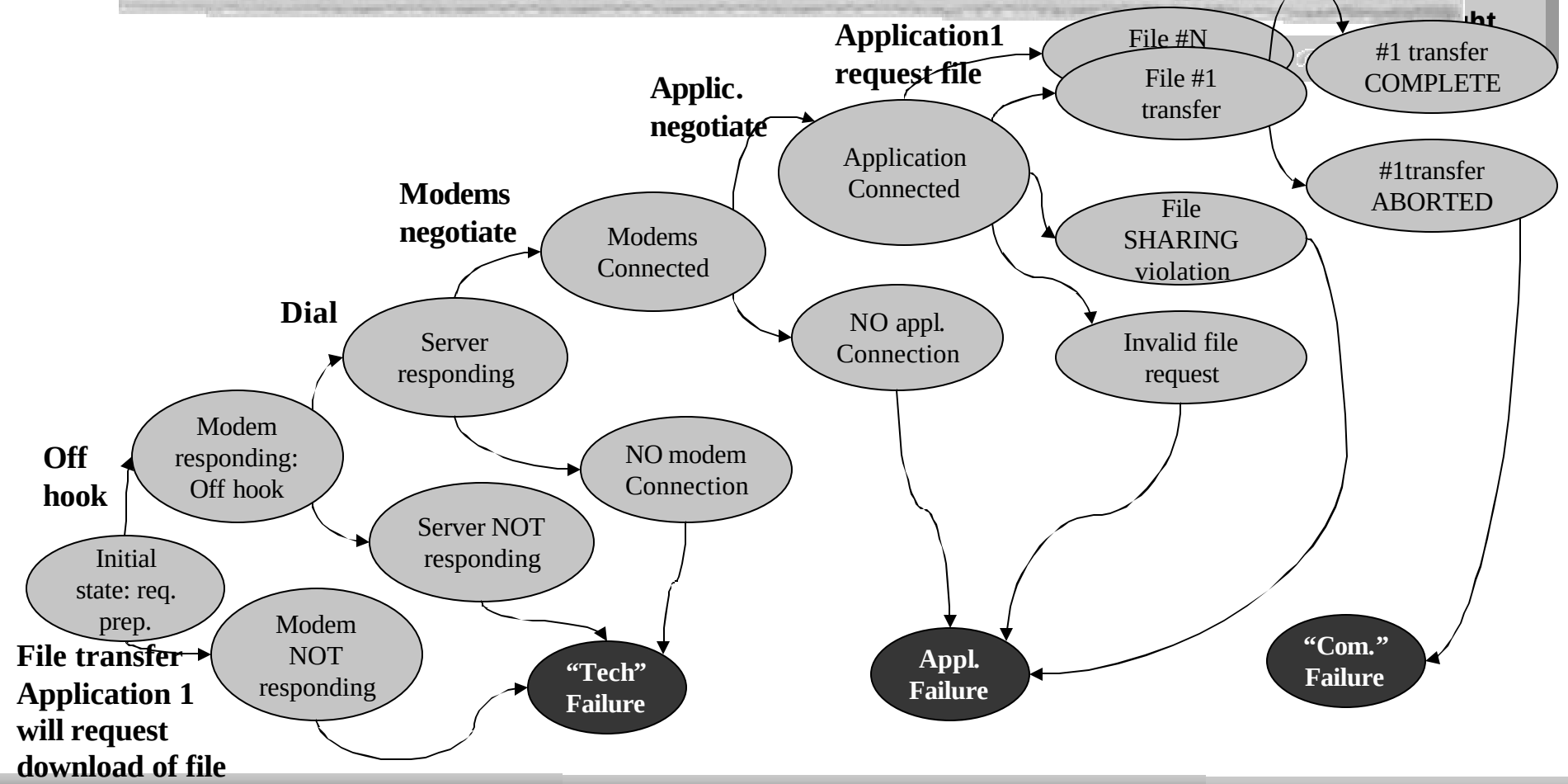
State Model-Based Testing

- Look at
  - All the possible inputs the software can receive, then
  - All the operational modes, (something in the software that makes it work differently if you apply the same input)
  - All the actions that the software can take
  - Do the cross product of those to create state diagrams so that you can see and look at the whole model
- Example:
  - Spent 5 hours looking at the API list, found 3 – 4 bugs, then spent 2 days making a model and found 272 bugs. The point is that you can make a model that is too big to carry in you head. Modeling shows inconsistencies and illogicalities.

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration: Models

State Transition Diagram – File Transfer



©2002 Amland Consulting

Styles of Exploration: Models

Failure Model

Whittaker: Why Software Fails (1999/2002)
"The Fundamental cause of software errors"

- Constraint violations
  - Input constraints
    - Such as buffer overflow
  - Output constraints
  - Computations
    - Look for divide by zeros and rounding errors. Figure out inputs that you give the system that will make it not recognize the wrong outputs
  - Data violations
  - Really good for finding security holes

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration: Failure Models

"How to Break Software" (1)

- Being a tester means finding bugs efficiently:
  - Set clear goals for every test case
  - Understand where bugs might hide
  - Know how to expose them
- The method:
  - Collect and study a large number of bugs in released products
  - Understand why they occur and what type of test it would take to find them
  - Generalize the test into "attack patterns" and teach students how to execute these patterns
  - Collect even more bugs, classify them and refine the attacks

How to Break Software, Whittaker et. al. 2000

©2002 Amland Consulting

Styles of Exploration: Failure Models

"How to Break Software" (2)

- Input Constraint Attacks:
  - Force all error messages to occur
  - Apply inputs that force default values
  - Explore character sets and data types
  - Overflow input buffer
  - Find inputs that may interact
    - Test various combinations of their values
  - Repeat the same inputs many times

©2002 Amland Consulting

Styles of Exploration: Failure Models

"How to Break Software" (3)

- Output Constraint Attacks
  - Force different outputs for each input
  - Force invalid outputs
  - Force output size change
  - Force output to exceed output space
  - Force the screen to refresh

©2002 Amland Consulting

Styles of Exploration: Failure Models

"How to Break Software" (4)

- Storage Constraint Attacks
  - Apply inputs under differing initial conditions
  - Data Structure Over/Underflow
    - Force a data structure to store too many or too few values
  - Find alternate ways to violate internal data constraints

©2002 Amland Consulting

Styles of Exploration: Failure Models

"How to Break Software" (5)

- Computation Attacks
  - Experiment with invalid operand and operator combinations
  - Force a function to call itself recursively
  - Force computation results to be too large or too small
  - Find features that share data or interact poorly

©2002 Amland Consulting

Exercise 3

- Select a different area of StarOffice (or continue where you left, if you prefer and are productive), or
- Select defect handling in MiniTest

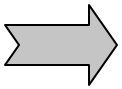
- Create a charter (including a mission) and select a (different) testing style and continue to test the AUT

©2002 Amland Consulting

Styles of Exploration:

Examples

- Use Cases
- Simple Walkthroughs
- Positive Thinking
- Scenarios
- Soap Operas



From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Use Cases

- List the users of the system
- For each user, think through the tasks they want to do
- Create test cases to reflect their simple and complex uses of the system

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration: Example

For each Use Case verify…

- Sufficient tests, positive and negative, have been identified for each flow of events (for the use cases that traverse your target-of-test)
- Tests to address any business rules implemented by the use cases, ensuring that there are tests inside, outside, and at the boundary condition / value for the business rule
- Tests to address any sequencing of events or actions, such as those identified in the sequence diagrams in the design model, or user interface object states or conditions.
- Tests to address any special requirements defined for the use case, such minimum/maximum performance, sometimes combined with minimum/maximum loads or data volumes during the execution of the use cases.

Modified from Rational Unified Process (RUP), ©Rational
©2002 Amland Consulting

Simple Walkthroughs

- Test the program broadly, but not deeply.
  - Walk through the program, step by step, feature by feature.
  - Look at what's there.
  - Feed the program simple, non-threatening inputs.
  - Watch the flow of control, the displays, etc.

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration: Example

Positive Testing

- Try to get the program working in the way that the programmers intended it.
- One of the points of this testing is that you educate yourself about the program. You are lookin at it and learning about it from a sympathetic viewpoint, using it in a way that will show you what the value of the program is.
- This is true “positive” testing – you are trying to make the program show itself off, not just trying to confirm that all the features and functions are there and kind of sort of working.

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration: Example

Scenarios

- The ideal scenario has several characteristics:
  - It is realistic (e.g. it comes from actual customer or competitor situations)
  - There is no ambiguity about whether a test passed or failed
  - The test is complex, that is, it uses several features and functions
  - There is a stakeholder who will make a fuss if the program doesn't pass this scenario

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration: Example - Scenarios

Soap Operas

- Build a scenario based on real-life experience. This means client / customer experience.
- Exaggerate each aspect of it:
  - Example, for each variable, substitute a more extreme value
  - Example, if a scenario can include a repeating element, repeat it lots of times
  - Make the environment less hospitable to the case (increase or decrease memory, printer resolution video resolution, etc.)
- Create a real-life story that combines all of the elements into a test case narrative.

Cem Kaner referensing Hans Buwalda (2001)

©2002 Amland Consulting

Styles of Exploration:

Invariances

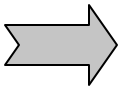
- Making changes that should NOT affect the program.
- Examples:
  - Sending text and graphics in different orders to a printer
  - Using VERY large files with programs that should handle large files
  - Mathematical operations in different but equivalent orders

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration:

Interference

- Interrupt
- Change
- Stop
- Pause
- Swap
- Compete



From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration: Interference

Interrupt

- Generate interrupts
  - From a device related to the task (e.g. pull out a paper tray, perhaps one that isn't in use while the printer is printing)
  - From a device unrelated to the task (e.g. move the mouse and click while the printer is printing)
  - From a software event

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration: Interference

Change

- Change something that this task depends on
  - Swap out a floppy
  - Change the printer that the program will print to (without signaling a new driver)
  - Change the video resolution

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration: Interference

Stop

- Cancel the task (at different points during its completion)
- Cancel some other task while this task is running
  - A task that is in communication with this task (the core task being studied)
  - A task that will eventually have to complete as a prerequisite to completion of this task
  - A task that is totally unrelated to this task

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration: Interference

Pause

- Find some way to create a temporary interruption in the task
- Pause the task
  - For a short time
  - For a long time (long enough for a timeout, if one will arise)
- Put the printer on local
- Put a database under use by a competing program, lock a record so that it can't be accessed – yet!

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration: Interference

Swap (out of memory)

- Swap the process out of memory while it is running (e.g. change focus to another application and keep loading or adding applications until the application under test is paged to disk
  - Leave it swapped out for 10 minutes or whatever the timeout period is. Does it come back? What si its state? What is the state of processes that are supposed to interact with it?
  - Leave it swapped out *much* longer than the timeout period. Can you get it to the point where it is supposed to time out, then send a message that is supposed to be received by the swapped-out process, then time out on the time allocated for the message? What are the resulting state of this process and the one(s) that tried to communicate with it?
- Swap a related process out of memory while the process under test is running.

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration: Interference

Compete

- Examples:
- Compete for a devise such as a printer
  - Put device in use, then try to use it from software under test
  - Start using device, then use it from other software
  - If there is a priority system for device access, use software that has higher, same and lower priority access to the device before and during attempted use by software under test.
- Compete for processor attention
  - Some other process generates an interrupt (e.g. ring into the modem, or a time-alarm in your contact manager)
  - Try to do something during heavy disk access by another process
- Send this process another job while one is underway

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration:

Error Handling

- Review possible error messages
  - Press the wrong key at the error dialog
  - Make the error several times in a row
- Device related errors
  - (disk full, printer not ready etc.)
- Data-input errors
  - Corrupt files, missing data, wrong data etc.
- Stress / Volume
  - Huge files, too many files, tasks, devices, fields, records etc.)

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration:

Troubleshooting

We often do exploratory tests when we troubleshoot bugs:

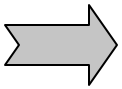
- Bug Analysis
  - Simplify the bug by deleting or simplifying steps
  - Simplify the bug by simplifying the configuration (or background tools)
  - Clarify the bug by running variations to see what the problem is
  - Clarify the bug by identifying the version that it entered the product
  - Strengthen the bug with follow-up tests (using repetition, related tests, related data, etc.) to see if the bug left a side effect
  - Strengthen the bug with tests under a harsher configuration
- Bug regression: vary the steps in the bug report when checking if the bug was fixed

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration:

Group Insight

- Brainstormed test lists
- Group discussion of related components
- Fishbone analysis
- Paired Exploratory Testing



From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration: Group Insight

Brainstormed Test Lists

Example (copy):

- Here is the program's specification:
  - This program is designed to add two numbers, which you will enter
  - Each number should be one or two digits
  - The program will print the sum. Press *Enter* after each number
  - To start the program, type ADDER
- Before you start testing, do you have any questions about the spec?

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration: Group Insight

Brainstormed Test Lists

- Summary of Example:
  - You brainstormed a list of tests for the two-variable, two-digit problem:
    - The group listed a series of cases (test case, why)
    - You then examined each case and the class of tests it belonged to, looking for a more powerful variation of the same test.
    - You then ran these tests.
  - You can apply this approach productively to any part of the system.

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration: Group Insight

Group Discussion of Related Components

- The objective is to test the interaction of two or more parts of the system
- The people in the group are very familiar with one or more parts. Often, no one person is familiar with all of the parts of interest, but collectively the ideal group knows all of them.
- The group looks for data values, timing issues, sequence issues, competing tasks, etc. that might screw up the orderly interaction of the components under study.

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration: Group Insight

Fishbone Analysis

- Fishbone analysis is a traditional failure analysis technique. Given that the system has shown a specific failure, you work backwards through precursor states (the various paths that could conceivably lead to this observed failure state).
- As you walk through, you say that Event A couldn't have happened unless Event B or Event C happened. And B couldn't have happened unless B1 or B2 happened. And B1 couldn't have happened unless X happened, etc.
- While you draw the chart, you look for ways to prove that X (whatever, a precursor state) *could* actually have been reached. If you succeed, you have found one path to the observed failure.

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Paired Exploratory Testing

- See previously presentation on testing in pairs, section 3.2 – Exploratory Testing in Pairs.
- Developed independently of paired programming, but many of the same problems and benefits apply.
- The eXtreme Programming community has a great deal of experience with paired work and offers many lessons:
  - Kent Beck, *Extreme Programming Explained*
  - Ron Jeffries, Ann Anderson & Chet Hendrickson, *Extreme Programming Installed*
- Laurie Williams of NCSU does research in pair programming. For her publications, see http://collaboration.csc.ncsu.edu/laurie/

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration:

Specifications

- Active reading – Tripos
- Active reading – Ambiguity analysis
- User Manual
- Consistency Heuristics

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration: Specifications

Active Reading – Developing Questions

- Satisfice Testing Model:
  - When you run out of testing ideas, walk the chart looking for a project / product / quality factor that you haven't based a test on recently
  - Randomly combine project / product / quality factors – make up a test case that is influenced by the selected product factor, that test the selected product element against the selected quality criterion.
  - Analyze a specification, operating on the assumption that every statement defines a project factor, a product factor or a quality criterion.

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration: Specifications

Active Reading (Ambiguity Analysis)

- There are all sorts of sources of ambiguity in software design and development
  - In the wording or interpretation of specifications or standards
  - In the expected response of the program to invalid or unusual input
  - In the behavior of undocumented features
  - In the conduct and standards of regulators / auditors
  - In the customers' interpretation of their needs and the needs of the users they represent
  - In the definitions of compatibility among 3rd party products
- Whenever there is ambiguity, there is a strong opportunity for a defect (at least in the eyes of anyone who understands the world differently from the implementation).

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration: Specifications

User Manual

- Write part of the user manual and check the program against it as you go.
- Any writer will discover bugs this way.
- An exploratory tester will discover quite a few this way.

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

Styles of Exploration: Specifications

Consistency Heuristics

- Discussed previously: HICCUPP
  - Consistent with History: Present function behavior is consistent with past behavior.
  - Consistent with our Image: Function behavior is consistent with an image that the organization wants to project.
  - Consistent with Comparable Products: Function behavior is consistent with that of similar functions in comparable products.
  - Consistent with Claims: Function behavior is consistent with what people say it's supposed to be.
  - Consistent with User's Expectations: Function behavior is consistent with what we think users want.
  - Consistent within Product: Function behavior is consistent with behavior of comparable functions or functional patterns within the product.
  - Consistent with Purpose: Function behavior is consistent with apparent purpose.

From Black Box Software Testing, copyright © 1996 – 2002 Cem Kaner
©2002 Amland Consulting

"I thought skill is the ability to something, more or less.

Skill varies from person to person. It's distinct from talent and knowledge.

A technique, by contrast, is a way of doing something; a sort of recipe.

Skill belongs to a person, technique is universal."

James Bach, Satisfice Inc.

James@satisfice.com
©2002 Amland Consulting

1.
2.
3.
4.
5.

Styles of Exploration Summary: Skills

- General systems modelling and dimensional analysis
- Inferencing out loud
- Experiment design
- Technical story telling
- Use of mnemonics and heuristics
- De-biasing (individual or team)
- Exploratory investigation
- Risk analysis
- Bug advocacy
- +++?

James@satisfice.com
©2002 Amland Consulting

1.
2.
3.
4.
5.

… and dimensional analysis

- …is the process of analyzing the dimensions of something.
- Example – dimensional analysis done on a wine glass:
  - volume of the glass
  - height of the glass
  - width at it's widest point
  - deviation from perfect circularity
  - mass
  - pieces (enumeration of distinct components: base, stem, cup)
  - melting point
  - material it's made from
  - age
  - surface it is currently standing upon
  - sentimental value
  - market value
- The skill of dimensional analysis is about modelling something in many different ways, any of which may figure in how w will test it

James@satisfice.com
©2002 Amland Consulting

Inferencing Out Loud

- The ability to talk through a logical progression either forward from evidence to conjecture, or backward from conjecture to evidence
- Using the HICCUPP-heuristics:
  - Consistent with History: Present function behavior is consistent with past behavior.
  - Consistent with our Image: Function behavior is consistent with an image that the organization wants to project.
  - Consistent with Comparable Products: Function behavior is consistent with that of similar functions in comparable products.
  - Consistent with Claims: Function behavior is consistent with what people say it's supposed to be.
  - Consistent with User's Expectations: Function behavior is consistent with what we think users want.
  - Consistent within Product: Function behavior is consistent with behavior of comparable functions or functional patterns within the product.
  - Consistent with Purpose: Function behavior is consistent with apparent purpose.

James@satisfice.com
©2002 Amland Consulting

Styles of Exploration Summary: Skills

- General systems modelling and dimensional analysis
- Inferencing out loud
- Experiment design
- Technical story telling
- Use of mnemonics and heuristics
- De-biasing (individual or team)
- Exploratory investigation
- Risk analysis
- Bug advocacy
- +++?

James@satisfice.com
©2002 Amland Consulting

Heuristics (and rules) and Skills

"… we relate to heuristics as a tool to apply; something that might help us do the right thing in a given situation, whereas we relate to a rule as something to comply with; something that defines right behavior.

Using heuristics properly requires that you exercise discretion and judgment, on some level; whereas judgment may get in the way of rules. It's helpful to have contradictory heuristics, because that's like having a variety of advice available before making a decision; whereas contradictory rules make compliance impossible.

James@satisfice.com
©2002 Amland Consulting

Exercise 4

- Select a different area of StarOffice or MiniTest (or continue where you left off if you prefer and are productive), or
- Select WEB site: www.amland.no/et_test

- Create a charter (including a mission) and select a (different) testing style and continue to test the AUT

©2002 Amland Consulting

1.
2.
3.
4.
5.

Styles of Exploration Summary

Introduction

Test Management and Techniques

ET Planning, Exec. and Documentation

ET Styles

ET Management

- **Hunches**
- **Models**
- **Examples**
- **Invariances**
- **Interference**
- **Error handling**
- **Troubleshooting**
- **Group Insight**
- **Specifications**

©2002 Amland Consulting